CAPABILITIES OF MODERN DATABASES FOR STORING DATA IN THE MINING INDUSTRY

Angel Dimitrov

University of Mining and Geology “St. Ivan Rilski”, 1700 Sofia; E-mail: a_d_dimitrov@abv.bg

ABSTRACT. The article examines technologies that store and process vast arrays of data in the mining industry. The aim of the study is to make a comparative analysis through an experiment between the two main paradigms - relational database management systems and NoSQL data warehouse management systems. As a result of the experiments, conclusions have been made about the applicability of the DBMS types in the processing of huge data sets and guidelines are given both for additional research and in practical terms.

Key words: big data, database, DBMS, NoSql.

ВЪЗМОЖНОСТИ НА СЪВРЕМЕННИТЕ БАЗИ ДАНИ ЗА СЪХРАНЕНИЕ НА ДАННИ В МИННАТА ПРОМИШЛЕНОСТ

Ангел Димитров

Минно-геоложки университет „Св. Иван Рилски“, 1700 София

РЕЗЮМЕ: Статията разглежда технологиите, чрез които се записват и обработват големи масиви от данни в минната индустрия. Целта на изследването е да се направи сравнителен анализ чрез експеримент между двете основни парадигми – системи за управление на релационни бази данни и системи за управление на NoSQL хранилища за данни. В резултат на експериментите са направени заключения относно приложимостта на типовете СУБД при обработка на огромни масиви от данни и се дават насоки както за допълнителни изследвания така и в практически аспект.

Ключови думи: големи данни, бази данни, СУБД, NoSql.

Въведение

Минната индустрия е изправена пред редица предизвикателства, които насърчават внедряването на нови технологии и така наречените *големи данни*, които се генерират вследствие на ускоряващия се напредък на дигитализацията и са една от обещаващите технологии, които могат да променят изцяло минно-добивната индустрия.

Въпреки, че и в момента се правят опити за добиване и използване на големи данни в минната индустрия, все още съществуват редица проблеми, които са свързани с фундаменталните проблеми на самите големите данни и особено тяхното управление (УГД – Управление на Големи Данни в превод на английски - BDM).

Тази статия има за цел да запълни една от празнините, като представи основите на УГД в частта за обработка и съхранение като се прави кратко въведение в големите данни и УГД и се обсъждат предизвикателствата, с които се сблъсква минната индустрия в опитите да внедрява системи, които работят с големи данни.

В статията накратко се обобщават източниците на данни в минната индустрия и се представят потенциалните ползи от използването им за ефективното управление на минната индустрия. Освен това са представени експериментални изследвания за съхранение и обработка на големи данни в два типа системи за управление на бази данни, а именно релационни и нерелационни данни.

Изследването в статията има своята значимост при определяне на технологичните основи за УГД в минната индустрия като дава доказателства за предимствата на нерелационните хранилища при съхранение и обработка на огромни обеми от данни, генерирани в следствие на автоматизираните процеси в мините.

Заключенията в статията са предназначени както за изследователите, които проектират и разработват решения в областта на УГД в минното дело, така и за специалистите в областта на минния добив със своите практически изводи.

Изложение

Големи данни в индустрията

Индустрия 4.0 интегрира интернет на нещата (IOT) с индустриални техники през двадесет и първи век, позволявайки на свързани устройства да обменят данни, да ги интерпретират и използват за насочване на интелигентно поведение.

Дигитализацията на комуникациите и транспорта и дори някои процеси като пандемията от COVID 19 катализират дигитализацията на индустриалното развитие с модерни технологии, включително роботика, изкуствен

интелект, усъвършенствани материали, добавена реалност и много други.

Според широкото определение, данните са набор от събрани параметри за дадена система, които са организирани и запазена по начин, позволяващ обработването им с лекота с цел интерпретация, прогнози и анализ. Тези анализи са ценна информация за системата и ако бъдат обработени научно, биха могли да допринесат за ефективно управление на самата система.

В днешно време дигитализацията на всички етапи на индустриалното производство води до огромен бум на данни, генерирани от различни електронни сензори, компютри, видео потоци и много други. Този процес и резултатите от него са известни като *големи данни*.

По-ниските бариери за нови субекти, навлизащи във всяка индустрия, причинени от глобализацията и цифровизацията, водят до експоненциален растеж на конкуренцията. Това е причината компаниите да търсят нови възможности за оптимизиране на своите процеси, чрез ефективните подходи на управление, използвайки най-съвременни технологии както и е основен двигател на технологичната еволюция.

Генериране на данни

Според неотдавнашно проучване на General Electric (Shefali, 2017) промишлените данни отдавна са се превърнали в големи индустриални данни, като обемът им се увеличава с всяка изминала минута. Според проучването само от една машина, която произвежда бебешки продукти, се генерират около 152K данни в секунда, 9M данни за минута, 545M за час, и 4T данни за една година.

За ефективното управление на производството съществува огромна нужда от обработка на тези данни, за да се превърнат в информация и да се използват по конструктивен и предсказуем начин, генерирайки стойност на знанието и изграждайки цифрово доверие.

През 2015 г. е предложен работен документ относно принципите за представяне на данни в стандартите на Индустрия 4.0, където са идентифицирани два основни принципа: Кибер-физически системи (CPS) и Smart Factory (Mario, Tobias & Boris, 2015).

Във всяка CPS съществуват силни зависимости между физическите и софтуерните елементи, които функционират в различни пространствени и времеви измерения, показват различни поведенчески модели и комуникират помежду си по най-различни начини (https://www.nsf.gov/pubs/2010/ nsf10515/nsf10515.htm).

Цифровите близнаци са известни като фонова система (Martin, 2017) и са пример за консолидиран модел на физическо и виртуално устройство, което съхранява данни с цел конфигуриране на различни действия и реагиране на ситуации, за да се избегнат потенциални усложнения и да се повиши производителността където CPS играе важна роля.

Интелигентната фабрика е описана като фабрика, която използва контекстно-ориентирана технология, за да помогне на хората и машините да изпълняват задачите си по-ефективно (Mario, Tobias & Boris, 2015) и е ключова характеристика на Индустрия 4.0. Това е възможно, тъй като процесът на производство и създаване е интегриран и се насърчават най-добрите практики на всяко работно място в производството.

Интернет на нещата (IoT) и Интернет на услугите (IoS) са другите два съществени елемента на Индустрия 4.0. поради факта, че там се генерират данните, необходими за анализ и ефективно управление. Чрез Интернет на нещата се използват физически сензори и датчици, механични управляеми инструменти мобилни телефони, които чрез специални схеми за адресиране комуникират помежду си за постигане на зададените цели (Giusto, Iera, Morabito & Atzori, 2010).

В резултат на това тези процеси трябва да бъдат възможно най-гъвкави, за да се вземат независими решения за осигуряване на ефективност на услугата, а управлението на задачите се делегира до по-висок ранг само в случай на невъзможност за управление (Otto, 2014).

Съхранение и обработка на данни

Огромният обем от данни, които се генерират от различни датчици и сензори в минната индустрия, следва да се организира по такъв начин, че данните да бъдат достъпни своевременно и достоверни, за да могат да служат за управление на процесите и машините в мините за добив (Anastasova and Yanev, 2021).

Базите данни са основно базирани на компютърни системи. Система за управление на база данни (СУБД) е софтуерът, който позволява на всички автоматизирани информационни системи, с които се работи днес, да доставят данни до други приложения.

Основните типове бази данни са така наречените релационни бази данни и NoSQL бази данни. Появата на първите е още от зората на компютризацията на индустрията. Те се използват широко в минната индустрия: от програмите за счетоводство и контрол до различни системи за сигурност и комуникация (Yanev, 2019).

С появата на парадигмата IoT се оказва, обаче, че тези бази от данни съвсем не са перфектното хранилище за огромните обеми от данни, които се генерират всяка секунда от милиони сензори, датчици и системи за наблюдение в реално време.

Ограниченията на релационните бази данни за обработка на огромни масиви поставят началото на нов тип нерелационни СУБД, където данните не се съхраняват в табличен вид, а се клъстерират в облачни хранилища, което на практика прави хоризонталното и вертикално скалиране неограничени и за това са изключително подходящи за всякакви IoT решения (Dimitrov et al., 2021).

Сравнително изследване

В изследването са включени две релационни СУБД – MS SQL, и популярната релационна база данни с отворен код – MySQL. Резултатите от тестовете с MS SQL и MySQL са сравнени с NoSQL бази данни. NoSQL базите данни, които са включени в изследването са RedisDB и Cassandra.

Методология на изследването

Наборът от данни за оценка в експеримента е генериран от реално промишлено приложение, където се наблюдават множество машини с цел контрол на производителност и качество.

Всяка машина разполага с множество сензори, които отчитат техните стойности от различни гледни точки и тези стойности се зареждат и съхраняват в хранилището на данни. Наборът от данни е разделен според размера

съответно на 1ГБ, 2ГБ, 4ГБ, 6ГБ, на които съответстват 19 530 000, 37 800 000, 74,550,00 и 111 180 000 записа.

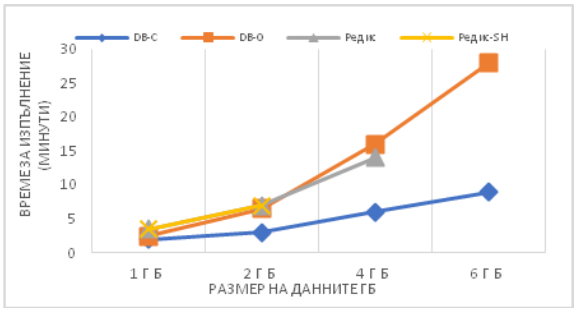
Първата заявка Q1 в експеримента е проста заявка за търсене на ключ и точно намиране на запис за дадена машина „m“, сензор „s“ и начален час „bt“. Втората заявка Q2, която се използва за бенчмарк в експеримента, включва избор на данни от хранилища по зададен диапазон. Заявката за агрегиране на данни Q3 има същите стойности за задаване на диапазон от Q2, но изчислителният метод на резултатите от заявката е в рамките на сървъра.

Тестовете в експеримента са проведени на машина с процесор Intel® CoreTM i5 с работна честота от 3.10ГХц x 4 и 64 битова операционна система Ubuntu 14.04 LTS. Машината разполага с 16ГБ оперативна памет и 500ГБ твърд диск.

За краткост са използвани акроними за всяка от базите данни. Обозначенията са: SQL-DB-C (MS SQL), SQL-DB-O (MySQL), КА (Касандра), КА-SH (Касандра с клъстер), КА-SH-R3-W (Касандра с клъстер и слаба последователност), КА-SH-R3-S (Касандра с клъстер и силна последователност), Редис и Редис-SH (Редис с клъстер).

Резултати от експеримента при групово зареждане

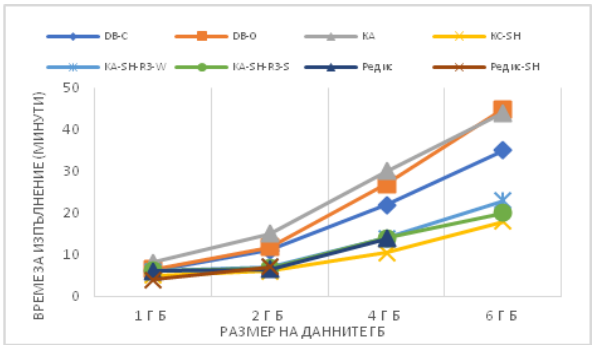
На фиг. 1 са показани резултатите от груповото зареждане без индексиране за DB-C, DB-O, Redis и Redis-SH сред четири размера набори от данни.



Фиг. 1. Резултат от групово зареждане

Като цяло DB-C се представя по-добре от другите в този експеримент, докато DB-O е по-бърза от Redis и Redis-SH между 1GB и 2GB, преди да започне да се мащабира в 4GB и 6GB диапазон. За двата типа Redis производителността е идентична и това е така, защото при Redis-SH шардовете не се разпределят паралелно.

Фигура **Фиг. 2**2 илюстрира изпълнението на масово зареждане с индекс на сензорен ключ с всички БД, участвали в този експеримент.

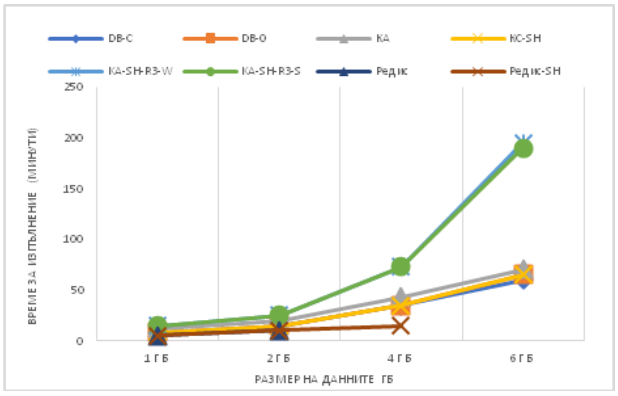


Фиг. 2. Групово зареждане с индекс сензорния ключ

При всички хранилища за данни Cassandra sharding (CA-SH, CA-SH-R3-W и CA-SH-R3-S) се наблюдава много по-ниско бързодействие в сравнение с другите поради липсата на едновременно разпределение на данни между фрагментите или възлите.

Redis и Redis-SH показват идентични резултати, както в предишния експеримент и се мащабират по-бързо от RDBMS и CA. DB-C, и DB-O се представят по-слабо, когато няма индексиране, което може да се дължи на зависимостта на разпределението на данните от стратегиите за индексиране.

От фиг. 3 става ясно, че всички БД предлагат мащабируема производителност на зареждане. Redis и Redis-SH са начело съответно до 4ГБ и 2ГБ. DB-O поддържа същата скорост колкото и DB-C до 74,550,000 (4ГБ) милиона необработени редове, но се представя по-бавно при 111,180,000 редове от данни.



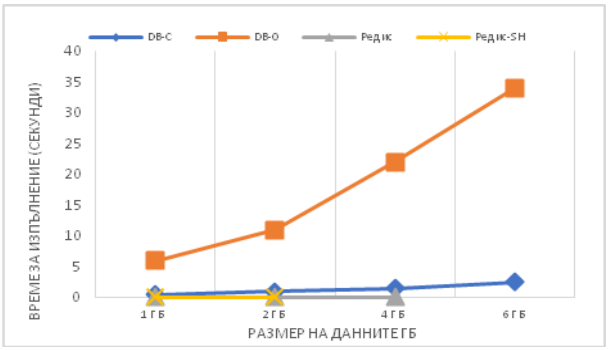
Фиг. 3. Групово зареждане с ключ за сензора и индекси на измерваните стойности

Характерно е, че DB-O, КА-SH-R3-W и КА-SH-R3-S се мащабират значително по-лошо от останалите, тъй като едва след зареждане на данните във всички възли и реплики се възстановява индексирането, което отне до 3 часа за зареждане на 6ГБ данни.

Важно е да се отбележи също така, че КА-SH-R3-W и КА-SH-R3-S имат идентична производителност, тъй като използват функцията за зареждане чрез SSTable, която ги зарежда директно в коректните възли на реплики.

Резултат от експеримента при проста селекция

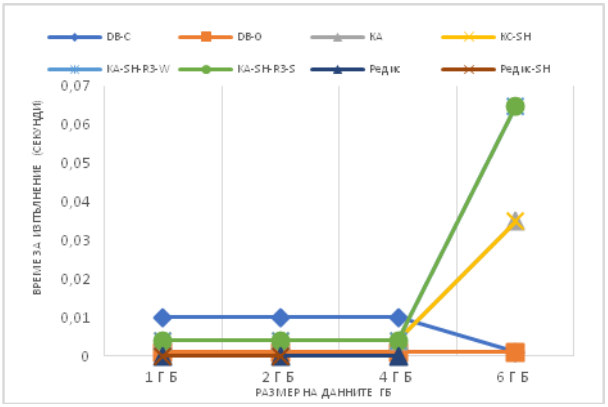
Резултатите от първия експеримент с проста заявка или търсене на ключ, който е проведен за четири системи DB-C, DB-O, Редис и Редис-SH, тъй като за останалите бази данни подобни заявки не са приложими е показан на фиг. 4.



Фиг. 4. Представяне при проста заявка без индексиране

И двете системи, които работят в паметта (Редис и Редис-SH) работят супер бързо, тъй като техните данни са готови в RAM и подредени по ключ-стойност. Въпреки че DB-C няма индексиране резултатите са противоположни на очакванията, тъй като пълното сканиране сред масивни редове от данни е бързо в сравнение с DB-O, като достига до повече от 32 пъти в сравнение с DB-C при 6 GB и около 21 пъти в сравнение Редис при 4 Гигабайта.

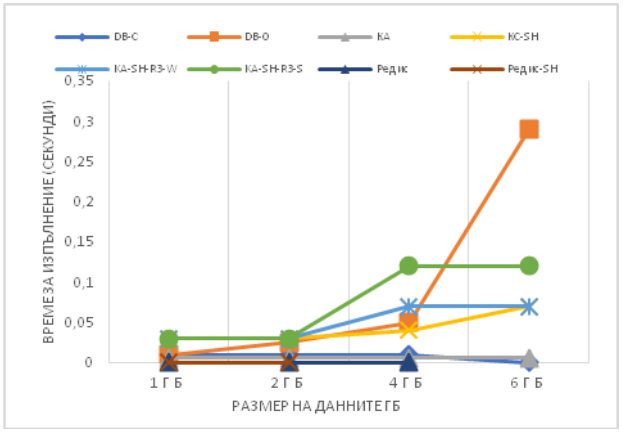
Във втория експеримент с търсене по ключове фиг. 5, индексирането е изградено върху съставен първичен ключ.



Фиг. 5. Представяне при проста заявка и индекс - ключ по колона сензор

Както се очаква, Редис и Редис-SH са най-бързите и имат същите резултати, тъй като Redis sharding пренасочва клиентите към възела, който има хранилище за ключ-стойност. Въпреки това в Касандра координаторът препраща заявката до възела, който има данните и изчаква отговора, след което отговаря на клиентите. RDBMS има по-добри резултати в повечето размери на наборите от данни от предишния експеримент и това се дължи на факта, че бързото извличане използва индекса на композитния ключ вместо пълното сканиране на таблицата.

Резултатите от третия експеримент са показани на фиг. 6 и се отнасят до търсене по ключове. Изградено е допълнително индексиране в колона „mv“.



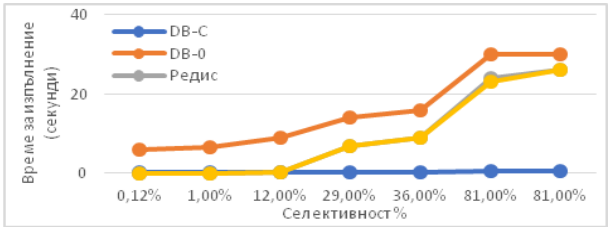
Фиг. 6. Представяне при проста заявка, ключ - колоната на сензора и индекси - колоните с измерените стойности

В резултат на този експеримент бързодействието на DB-C, KA, Редис и Редис-SH не е засегнато в сравнение с предишните експерименти, защото колоната, в която е изграден индексът, не се използва в заявката докато бързодействието на другите системи е засегнато отрицателно в този експеримент.

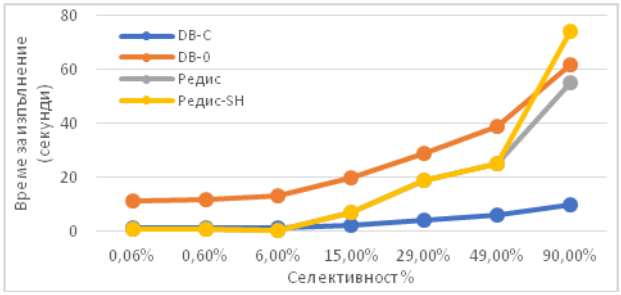
В Касандра създаването на индекс по колона MV не повлиява на стъпките на извличане на данни, тъй като данните са разделени (m, s, bt), а Касандра зависи от факта, че Q1 директно засяга възела със стойностите на заявката.

Резултат от експеримента при търсене в диапазон

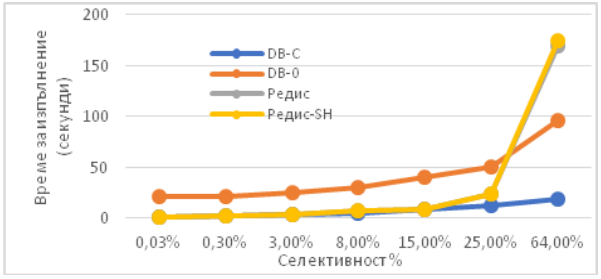
Първите експерименти със заявка за търсене на обхват (Q2) са проведени без индексиране за 1ГБ, 2ГБ, 4ГБ и 6ГБ обем на набори от данни. Както е показано на фигури 7, 8, 9 и 10, въпреки че не са използвани индекси, DB-C се представя по-бързо във всички експерименти в сравнение с други системи, докато DB-O е най-бавна във всички сценарии на експеримента.



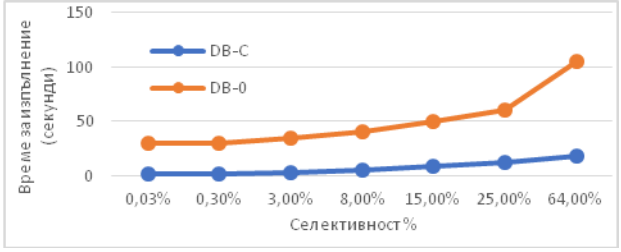
Фиг. 7. Представяне при заявка за диапазон без индексиране за 1ГБ



Фиг. 8. Представяне при заявка за диапазон без индексиране за 2ГБ



Фиг. 9. Представяне при заявка за диапазон без индексиране за 4ГБ



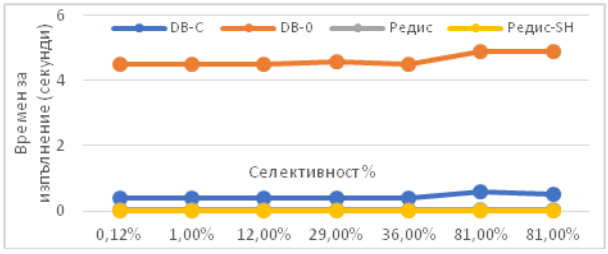
Фиг. 10. Представяне при заявка за диапазон без индексиране за 6ГБ

Резултатите от представянето на Redis и Redis-SH, обаче, са неочаквани, въпреки факта, че се използва структурата Sorted Set в заявката и резултатът от Q2 се съхранява сортиран в паметта.

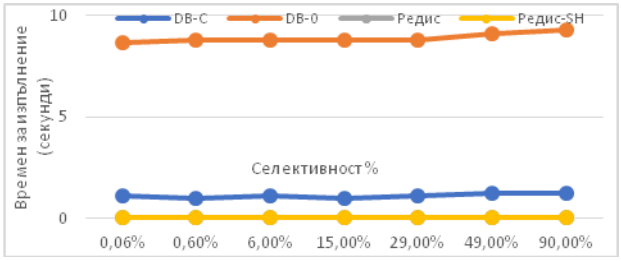
Извличането на данни за голям диапазон е по-лошо от колкото при DB-O, но обратното при по-нисък диапазон на селективност. Тъй като размерите на паметта в тези експерименти са 7ГБ, 13.5ГБ и 12.65ГБ (в част 2) при съответно 1ГБ, 2ГБ и 4ГБ набори от данни, това забавяне може да се дължи на пълното използване на паметта на устройството и така да не остава място за запитванията, прехвърлянето, размяната и буферирането на резултатите от наборите от данни.

Резултати при агрегирана заявка

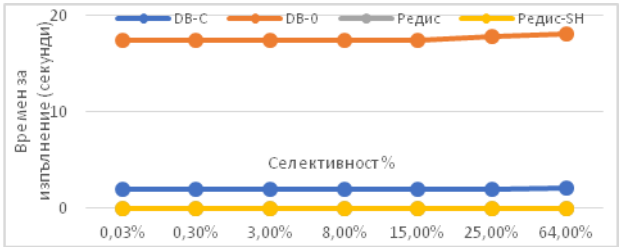
Резултатите от производителността на заявката за агрегиране (Q3) без индексиране за изследваните системи са илюстрирани на фигури 11, 12, 13 и 14.



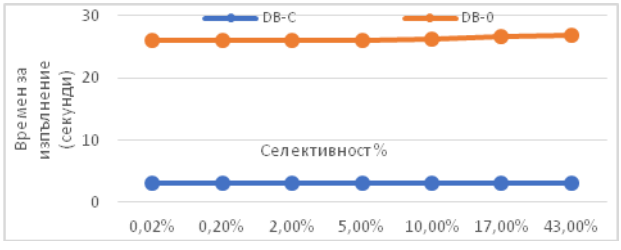
Фиг. 21. Представяне при агрегирана заявка без индексиране за 1ГБ



Фиг. 32. Представяне при агрегирана заявка без индексиране за 2ГБ



Фиг. 43. Представяне при агрегирана заявка без индексиране за 4ГБ



Фиг. 54. Представяне при агрегирана заявка без индексиране за 6ГБ

Използвани се същите диапазони на селективност както при заявката Q2, с изключение на API клиента, т.е. четенето е направено директно от системните обвивки.

Системите Cassandra са изключени от тези експерименти, тъй като те трябва да бъдат индексирани през цялото време чрез основна стратегия за индексиране, като например първичен ключ, който по -късно се използва като ключ за разделяне.

В този експеримент се установява, че и Редис, и Редис-SH се представят супер бързо в сравнение с предишните експерименти със заявка Q2, което може да се обясни с елиминирането на трансфера на данни между системата и API клиента.

Времето за изпълнение на всички заявки в набори от данни 1ГБ, 2ГБ и 4ГБ е незначително (0.1 милисекунда). Техните резултати, както обикновено, са идентични, тъй като е известно, че Редис пренасочва клиента на шела към възела или фрагментите, където се съхранява ключ-стойност.

Високата скорост на извличане на данни може да се обясни и със структурата на съхраняване на ключ-стойност в Sorted Set, където сортирането зависи от стойността „mv“. Ефективността на DB-C е на второ място в тези експерименти и скоростта е много по-висока в сравнение със заявката Q2 на същото ниво, особено при висок диапазон на селективност. Въпреки, че в това изследване няма API клиент, DB-O се мащабира по-лошо от другите системи, но по-добре от същата инстанция в заявката Q2.

Заключение

Резултатите от това изследване показват, че няма конкретен тип система, която да превъзхожда другите във всички сценарии и най-добрият вариант може да варира в зависимост от характеристиките на данните, вида на заявката и конкретната система.

И двете системи за съхранение на данни в паметта Редис и Редис-SH се представят добре в сравнение с всички останали системи при зареждане и анализ на постоянни регистрационни файлове.

DB-C също показва подобно добро представяне. Въпреки че RDBMS DB-O с отворен код също показва приемливи резултати, тази система не може да се конкурира със споменатите по-горе системи.

Всички системи Касандра показват сравнително добра производителност при масово зареждане на необработени данни с помощта на инструмента за групово зареждане (SSTable loader), чиято функция е да зарежда данните в клъстер от възли и да прехвърля съответната част от данните към всеки възел и реплика паралелно. За този сценарий, обаче, е характерно, че производителността се влошава когато вторичното индексиране е създадено преди масовото зареждане.

Всички системи на Касандра се представят слабо при заявките за агрегиране и обхват, но те са конкурентни при ключови заявки за търсене, извличащи определен запис. Използването на първичен и вторичен индекс в двете RDBMS води до добри резултати за търсене и съвпадение по конкретен ключ и за извличане на селективен диапазон от данни.

Създаването на индекси преди масовото зареждане има отрицателно въздействие върху производителността. Както Редис, така и Редис-SH нямат вградени първични или вторични индекси, но структурата на ключовете при тях може да се използва като достъп до първичен ключ, както в основната заявка за избор, където ключът всъщност е съставен ключ. В допълнение, структурите от данни на Редис могат да се използват за работа като вторични индекси, както във втората така и в третата заявка. В тези случаи ключовата структура идентифицира как са сортирани данните.

Системите с Касандра предоставят както първични индекси, така и вторични индекси, но за съжаление вторичният индекс на Касандра не осигурява пълно вторично индексиране, както в RDBMS. Това се нарича „полу-вторичен индекс“. Първичният индекс в Касандра е фундаментална структура от данни и е основният фактор, определящ ефективността на извличане на данни. За разлика от първичните ключове е установено, че вторичните индекси в Касандра оказват негативно влияние върху изпълнението както на задачи за зареждане, така и на заявки.

Макар че нивото на съгласуваност при четене и писане няма ефект върху КА и КА-SH, то влияе значително както върху КА-SH-R3-W, така и върху КА-SH-R3-S, където са използвани реплики. То обаче няма влияние върху зареждането на постоянни файлове при използване на инструмент за бърза миграция на данни, като помощната програма за зареждане SSTable.

За разлика от това, Редис остава последователен, тъй като пренасочва заявките на клиентите към възела ключ-стойност. В случай на репликация, Redis е по-малко последователен, но това няма ефект върху производителността поради асинхронното поведение на репликация, характеризиращо тази система.

Паралелизиране на данните над множество фрагменти или възли оказва значителен положителен ефект върху производителността на масово зареждане на постоянните регистрационни файлове в системите Касандра, поради поведението на паралелно разпределение на данните между възлите на клъстера.

Това обаче се отразява негативно на анализа и филтрирането на данни. От друга страна паралелизацията на Редис не влияе нито върху зареждането, нито върху анализа на регистрационните файлове.

В минната индустрия се работи с данни, които са многофакторни, от разнородни източници и с все по-голям обем. Съществува и известна доза несигурност на проучванията, което прави така, че е от изключителна важност за работните процеси информацията да бъде обработена и анализирана максимално бързо. Като се има предвид това, тенденцията бази данни от типа NoSQL да навлизат все повече в индустрията изглежда закономерна.

Като финална бележка, тези сравнителни експерименти са проведени с помощта на хардуер от среден клас. Би било интересно да се повторят същите експерименти на машини от висок клас с не по-малко от 64 GB памет за експерименти без sharding и 32 ГБ възли за sharding експерименти.

Освен това клиентските приложения могат да се изпълняват на самостоятелни машини, които да са различни от сървъра, управляващ системите за бази данни, за да се симулира по-реалистичен сценарий.

Литература

Anastasova Y., N. Yanev. 2021, Models and data quality in information systems applicable in the mining Industry. – *E3S Web Conf.,(Vol. 280, p. 08012). EDP Sciences, Second International Conference on Sustainable Futures: Environmental, Technological, Social and Economic Matters.* https://doi.org/ 10.1051/ e3sconf/202128008012.

Dimitrov A., N. Ivanova, R. Nesheva, N. Yanev, 2021, Preparation of Industrial Data for Implementation in Big Data, – *Annual of the University of mining and geology "St. Ivan Rilski" – София, Volume 64*, p. 151-155.

Giusto D, A. Iera, G. Morabito, L. Atzori. 2010. *The Internet of hings*. Springer.

Mario, H., P. Tobias, O. Boris. 2015. Design Principles for Industrie 4.0 Scenarios: A Literature Review. – *Review. Dortmund Technische Universität*.

Martin. 2017. Industry 4.0: Definition, Design Principles, Challenges, and the Future of Employment. – *Cleverism Magazine*.

Otto, H. 2014. Technik für die wandlungsfähige Logistik. – *Industrie*.

Shefali, P. 2017. The opportunity and potential of industrial big data. – *General Electric*. https://www.ge.com/digital/sites/default/files/download_assets/Unlocking-Business-Value-Through-Industrial-Data-Management-whitepaper.pdf

Yanev N., 2019, Contemporary approaches in storage and data processing. – *Journal of Mining and Geological Sciences, Vol. 62*, p. 96-99.

https://www.nsf.gov/pubs/2010/nsf10515/nsf10515.htm